NAG-1-1624

E21 X73 : FINAL REPORT

Genetic Adaptive Control for PZT Actuators

FINAL
7N-33-CR
OCIT

Jeongwook Kim, Shelley K. Stover and Vijay K. Madisetti

NASA/CR-95- 206816

05 7141

***Abstract*— A piezoelectric transducer (PZT) is capable of providing linear motion if controlled correctly and could provide a replacement for traditional heavy and large servo systems using motors. This paper focuses on a genetic model reference adaptive control technique (GMRAC) for a PZT which is moving a mirror where the goal is to keep the mirror velocity constant. Genetic Algorithms (GAs) are an integral part of the GMRAC technique acting as the search engine for an optimal PID controller.**

Two methods are suggested to control the actuator in this research. The first one is to change the PID parameters and the other is to add an additional reference input in the system. The simulation results of these two methods are compared.

Simulated Annealing (SA) is also used to solve the problem. Simulation results of GAs and SA are compared after simulation. GAs show the best result according to the simulation results. The entire model is designed using the Mathworks' Simulink tool.

***Keywords*— GAs, GMRAC, PID, PZT, Simulink**

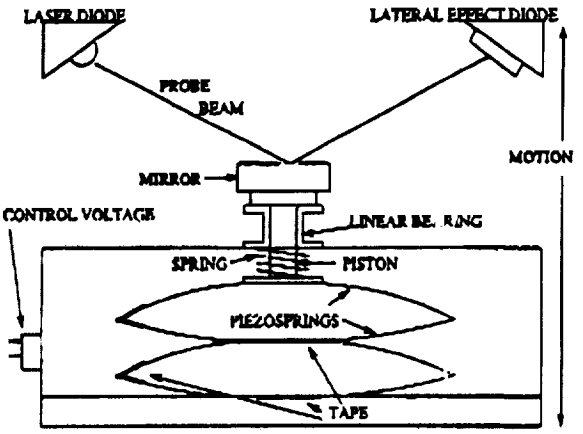

Figure 1. PZT System

I. Introduction

Most linear actuators are comprised of motors which are very heavy, but are capable of providing stable linear motion over long periods of time. Although the PZT actuator can provide a highly controllable linear motion, the inherent characteristics of the material can change due to aging and temperature. Therefore, an adaptive control technique is desirable to compensate for these changes. The GMRAC technique which incorporates genetic algorithms was used to set the PID parameters and thus control the velocity movement of a mirror being driven by the PZT.

Genetic Algorithms (GAs) have been useful in solving various kinds of problems. GAs are primarily used to solve optimization problems which have a complex, nonlinear search space. The design automation for controllers is one application for GAs. There are several instances where GAs have been applied to this type of problem: Lee and Takagi designed a fuzzy system using GAs for the inverted pendulum [5]; Michalewicz used GAs for solving the optimal control problem [7]; and Das, Goldberg, Maclay and Dorey used GAs for system identification [1, 6].

In this research, two different optimization approaches are proposed. The first is to change the PID gain parameters and the other is to adjust an additional input value. The performance of the two different methods are compared and simulated annealing (SA) is also used to compare the performance with GAs. According to the simulation results, the GA outperformed the other methods. Specifically, the first approach using GAs for PID gain control had the best result.

This research was supported in part by NASA Langley - Flight Electronics Division (1994-1995). Submission on 1 Oct 1995.

The authors are with the DSP Laboratory, Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0250, USA. vkm@ee.gatech.edu. http://www.ee.gatech.edu/users/215.

II. PZT System

The PZT system described here consists of stacks of piezoelectric structures which were invented at NASA's Langley Research Center. When an input voltage is applied to the PZT, a force is exerted by the piezoelectric which delivers a displacement to the attached mirror. These devices can be placed mechanically in series and connected electrically in parallel. The motion of each element adds in phase to produce the total motion which is almost equivalent to the sum of motion of each element. The system configuration is shown in Figure 1 [10]. The real photo image for this system is in Figure 2.

In this simulation, the PZT plant system was modeled in Simulink. The entire system model is shown in Figure 3.

III. GAs and GMRAC

According to evolution theory, nature will adapt to a changing environment in an efficient manner. The information about each individual is contained in its genes, and the processes of natural selection and progressive breeding build a strong population [2, 3]. Natural selection ensures that the genes from a strong individual are present in greater numbers in the next generation than those from

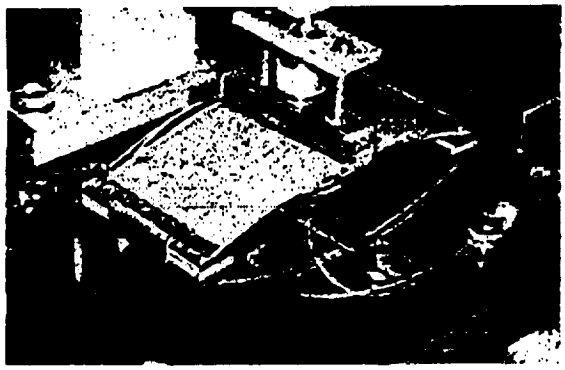

Figure 2. Photo Image for PZT System

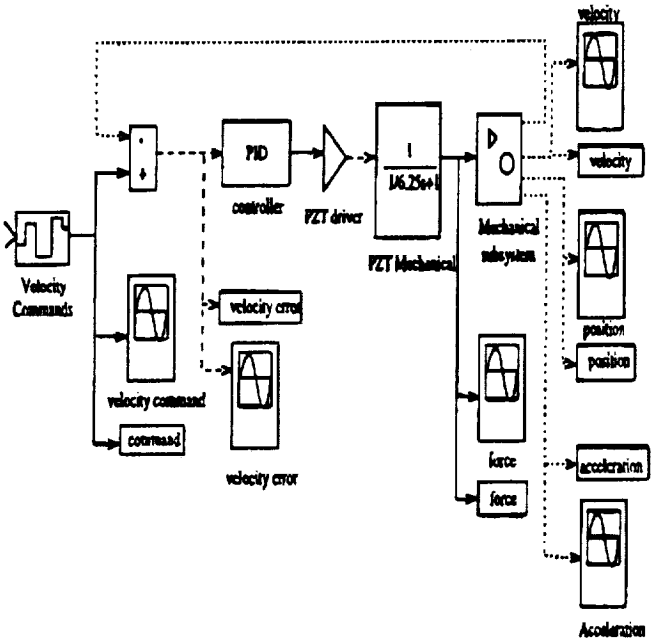

Figure 3. PZT Controller

a weak individual. After a number of generations, this process can combine stronger genes to produce an even stronger individual whose genes will strengthen the population. The *Genetic Algorithms* (GAs) are based on this *natural selection* and *genetics*. They combine survival of the fittest among string structures with an information exchange to form search algorithms with some of the innovative flair of human search. In every generation, a new set of strings is created from pieces of the old. GAs efficiently exploit historical information to determine a new search point with improved performance [4].

Complex structures can be coded by using simple representations such as bit strings [4]. These act like natural genes. A number of these strings or individuals form the population of GAs. In the next step, successive generations of individuals will be built by using genetic operators. These genetic operators are used to make new individuals

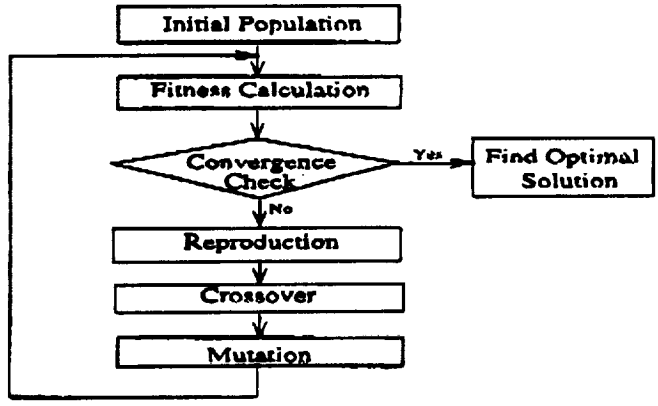

Figure 4. Basic Procedure of GAs

by combining the genes of other individuals. When this process is controlled properly, it can produce a rapid improvement in the overall fitness of the population. This produces a population of structures that is well-adapted to the problem. After number of generations, the structures represent good solutions: The fittest individual should be an optimal solution. GAs can perform the optimization process even in large and complicated search spaces. The basic procedure of GAs is shown in Figure 4. The most popular method of representing the problem is a binary bit string. In this string, we can use either '0' or '1' as a bit value. GAs using this method are known as *Binary-Coded Genetic Algorithms* (BCGA) [8].

Choosing the *fitness* evaluation for GAs is a very important factor. The fitness function must be devised for each problem to be solved. Given a particular string, the fitness function returns a single numerical "fitness". For many problems, particularly function optimization, it is obvious what the fitness function should measure. But this is not always the case. It is necessary to calculate the fitness of all individuals in an entire population. Therefore, it is better to use as simple a function as possible.

The GMRAC uses GAs with the plant model [9]. Receiving plant output and reference input information, GAs select the best PID gains at each time step from the candidate populations. It is very important for the adaptation to occur between one time step and the next. After a set number of time steps, a new output is generated using the best set of PID gains from the simulate time steps. Therefore, the process may not converge before the output must be set, but a near optimum value is selected for best performance.

The block diagram for the GMRAC is shown in Figure 5. GMRAC needs two types of plants, the real system and the plant model are needed for the implementation. In the simulation stage, the PZT simulink model represents both systems. A disturbance is used to simulate the real environment and adds noise to the system.

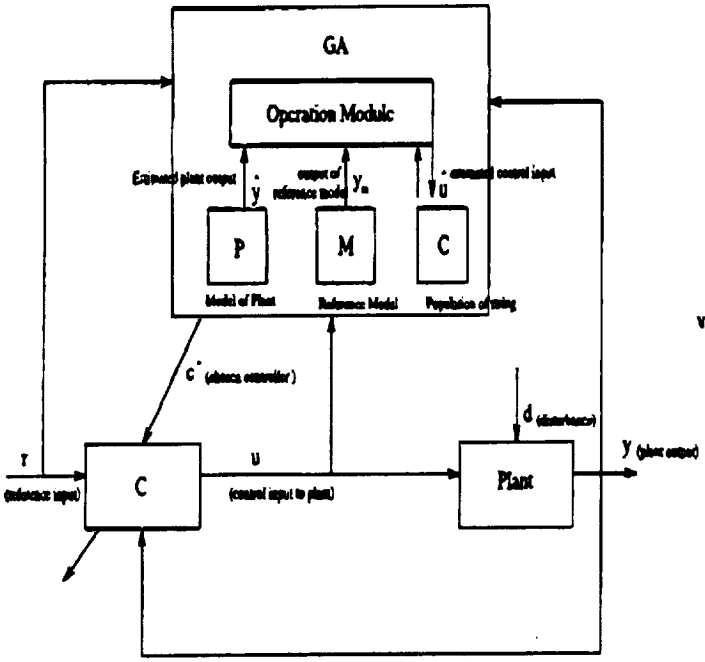

Figure 5. GMRAC

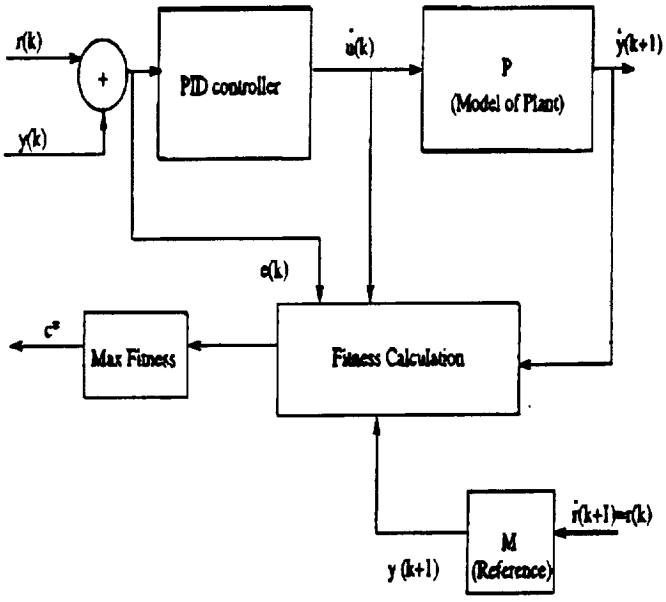

Figure 6. GMRAC Operation Module for PZT Control

IV. GA IMPLEMENTATION

The GMRAC operation module for the PZT system is shown in Figure 6. In the simulation, two types of optimization methods are used. The first one is to find the optimal set of PID gains. The other is to find an additional reference input to the error signal. The first method involves finding four gain factors but the second method just needs one value. Therefore, the size of the chromosome string is smaller in the second case. The simulation is performed for both methods. The model of the first case

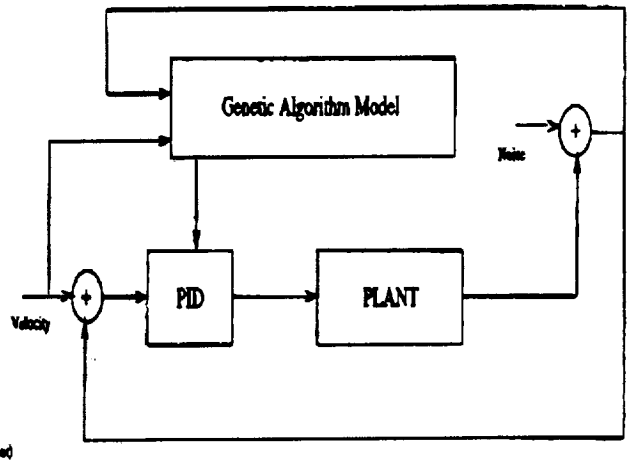

Figure 7. System for PID Gain Control

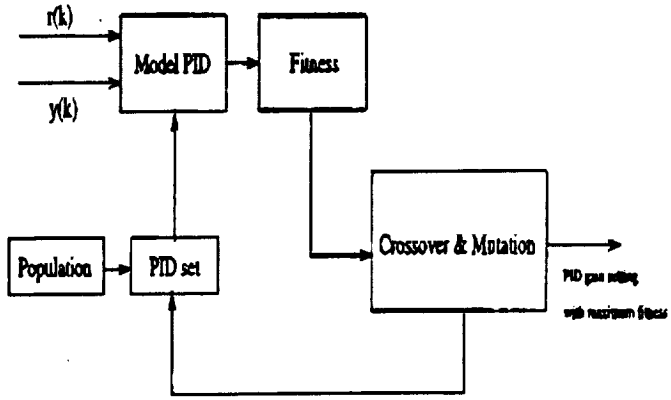

Figure 8. GA with PID

is shown in Figure 7. In this model, the GMRAC finds the best set of PID gains at each time step. In this simulation, the population size was set to 20. Each gain value is mapped into one string, and there are four parameter variables to be optimized: proportional, integral, derivative and derivative divisor parameters. Each of these gain elements is assigned a 6 bit value. The method of calculating the fitness is explained in section V. The crossover operation and mutation are used as genetic operators, linear scaling is used for preventing early convergence, and elite passing is used to keep the best string during the entire process.

The detailed block diagram of the GMRAC module for the PID gain optimization is shown in Figure 8. In this module, the fitness function gives the best fitness values to the set of PID gains which have the smallest difference between the reference input and the system output.

The second case is shown in Figure 9. In this case, the additional reference input is added to the reference input part in the original system. The size of string is normally smaller than when using PID gain control and thus reaches the optimal value much faster. The GMRAC module for this case is shown in Figure 10.

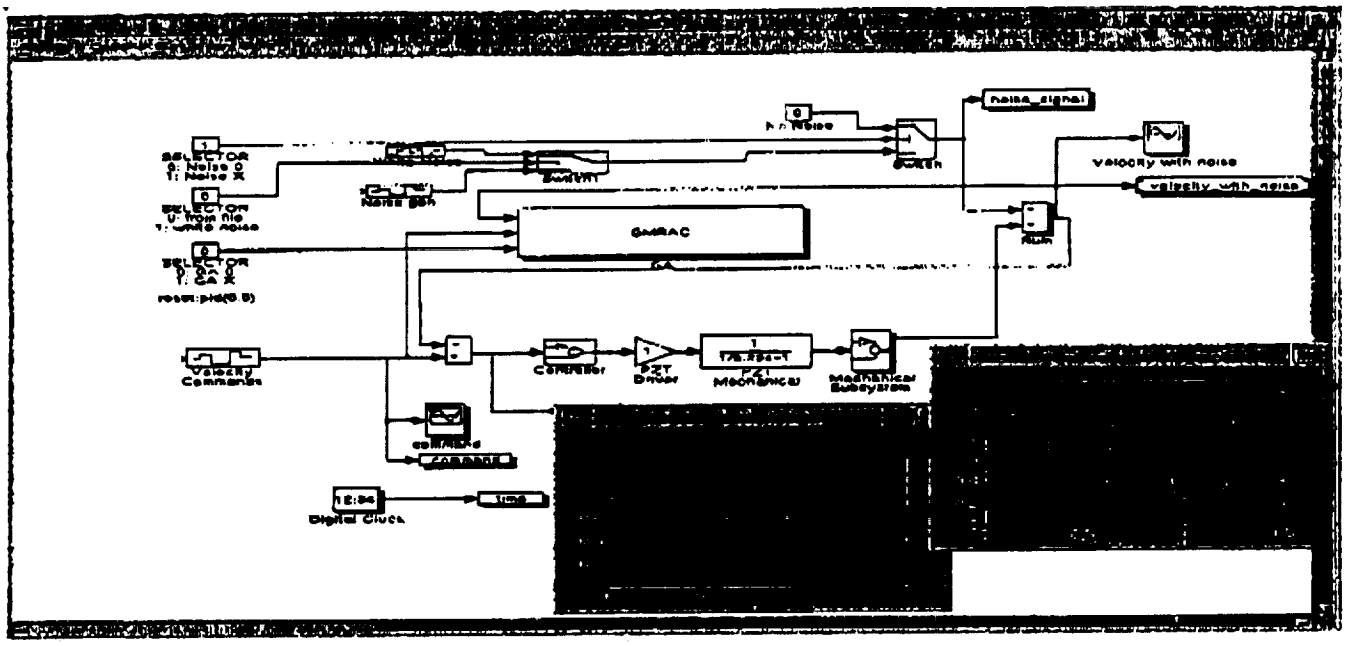

Figure 11. Simulation Environment

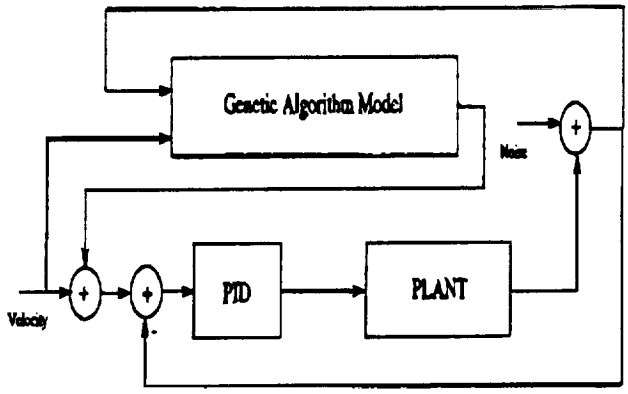

Figure 9. System for Additional Input

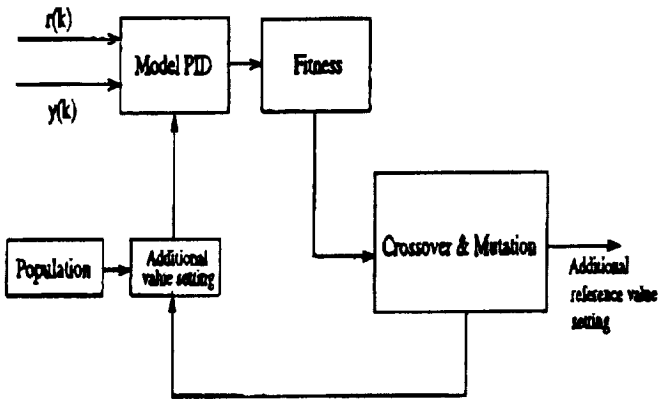

Figure 10. GMRAC Module for Additional Input

V. Simulation Result

The environment for simulation is shown in Figure 11. In Figure 11, there is a subblock for the GMRAC routine. Two output graphs are shown bottom right side. The simulation time and method can be decided by the simulation menu from the menu bar in the simulator.

Four types of fitness functions were used to get the best result. The definition of each of the fitness functions are as follows:

- $e(k) = y_m(k+1) - \hat{y}(k+1)$, $e_d(k) = e(k) - e(k-1)$
- $p_1 = e_d(k)$, $p_2 = e(k)$
- type 1: p_2
- type 2: $50p_1 + 500p_2$
- type 3: $500p_2$
- type 4: $500p_1 + 50p_2$
- Fitness : $\frac{1}{each\ type}$

The objective is to reduce the error. Therefore, the optimization value must have the smallest error. The minimization problem is easily converted to the maximization problem. This minimization problem is changed to the maximization problem by using the fitness value as $\frac{1}{error}$. The definition of each case is summarized in Table 1. Cases 1 to 8 use the PID gain control. Case numbers which are greater than 20 use the additional reference input control. Three input sets are used to compare the performance between these two methods. The input value starts at "0" and reaches "1" at 1 msec in the first case. The first 18 simulation results use this step input command. Simulation results with different fitness function types are shown in Figure 12. As shown in table 1, the noise is not added in these cases. The next simulations are for additional reference input control. The same types of fitness functions are used in this method, and the results are shown in Figure 13. The simulation results are almost the same for both methods. The fitness type does not affect the simulation result in these cases.

Type of Cases	Fitness Type	Noise
Case 1	fitness 1	No-noise
Case 2	fitness 2	No-noise
Case 3	fitness 3	No-noise
Case 4	fitness 4	No-noise
Case 5	fitness 1	With-noise
Case 6	fitness 2	With-noise
Case 7	fitness 3	With-noise
Case 8	fitness 4	With-noise
Case 9	No GAs	No-noise
Case 10	No GAs	With-noise
Case 21	fitness 1	No-noise
Case 22	fitness 2	No-noise
Case 23	fitness 3	No-noise
Case 24	fitness 4	No-noise
Case 25	fitness 1	With-noise
Case 26	fitness 2	With-noise
Case 27	fitness 3	With-noise
Case 28	fitness 4	With-noise

Table 1. Definition of Each Case

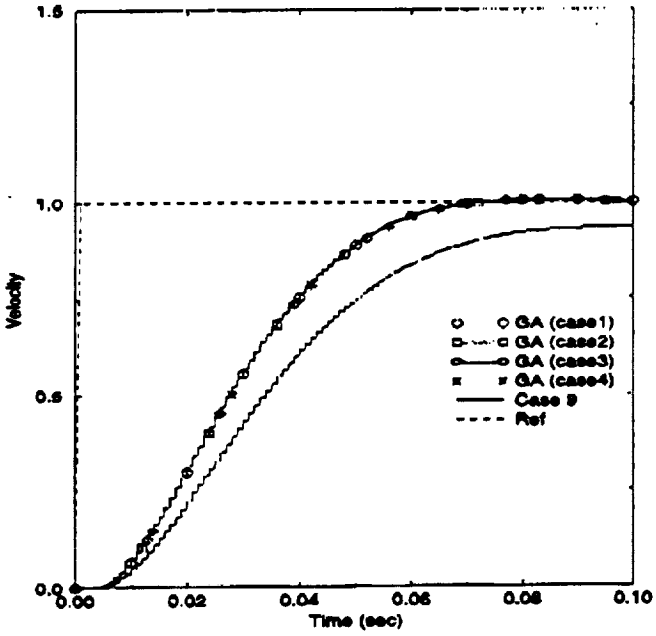

Figure 12. Simulation Result (case 1-4)

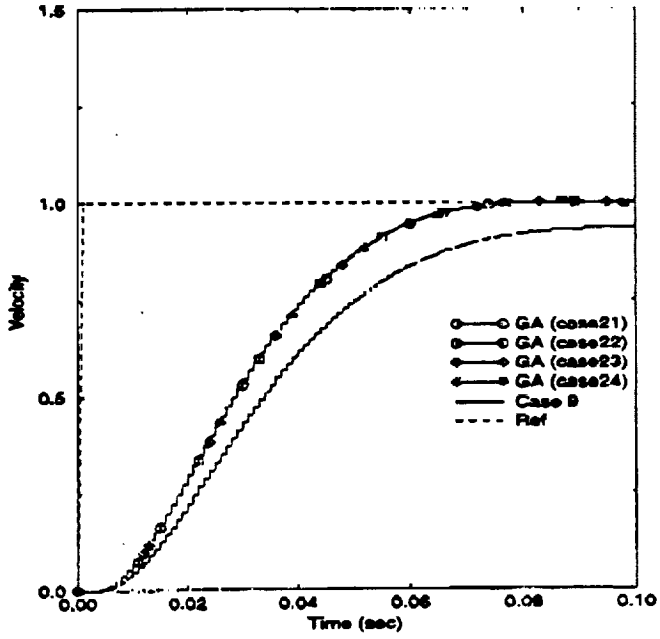

Figure 13. Simulation Result (case 21-24)

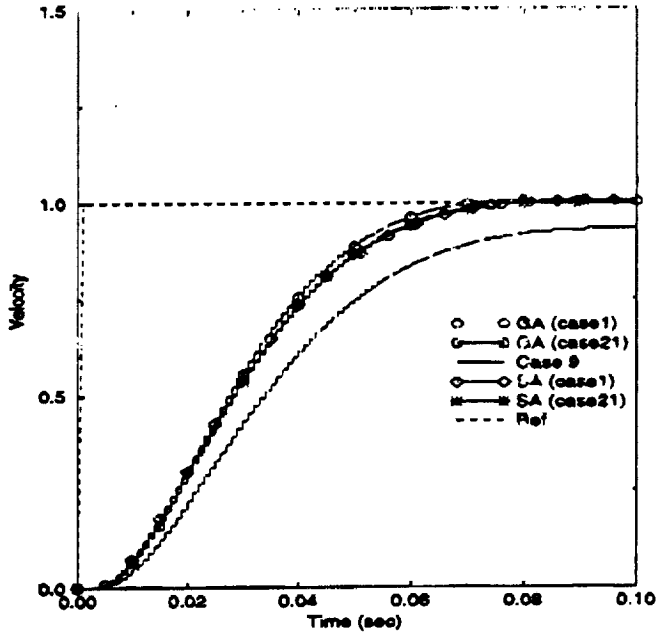

Figure 14. Simulation Result (GAs and SA)

The output results of GAs and SA with type 1 fitness are shown in Figure 14. GAs are not applied to case 9 and noise is not added. The simulated Annealing (SA) technique was also simulated for comparison using the same cases. Simulation results where white noise is added to the system are shown in Figure 15.

The second set of simulation uses a different input set. The input velocity command starts at value "1" and keeps this value to the end of simulation. In this case, the white noise is not used. The noise value starts with "0" at 0 sec and is changed to "0.1" at 0.15 sec, "-0.1" at 0.2 sec, and finally "0" at 0.35 sec. The algorithm uses the fitness type 1. The simulation result without noise is shown in Figure 16. The results with noise are shown in Figure 17. In this case, the type of fitness function affects the simulation results slightly. The fitness function type 1 provides the best result when compared against the others.

The final simulation uses a step function as an input signal. The result is shown in Figure 18. In the Figure, section (A) shows the result for the entire region. Sections (B),(C), and (D) show detailed results for specified regions in section (A). In this simulation, GAs and SA are used to solve the problem and four different results emerge from the simulation. The legend in the graph is explained as

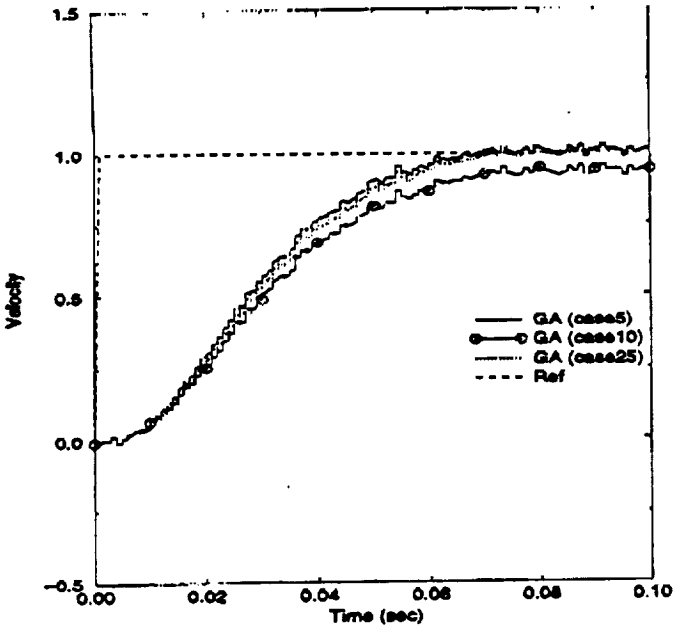

Figure 15. Simulation Result with Noise

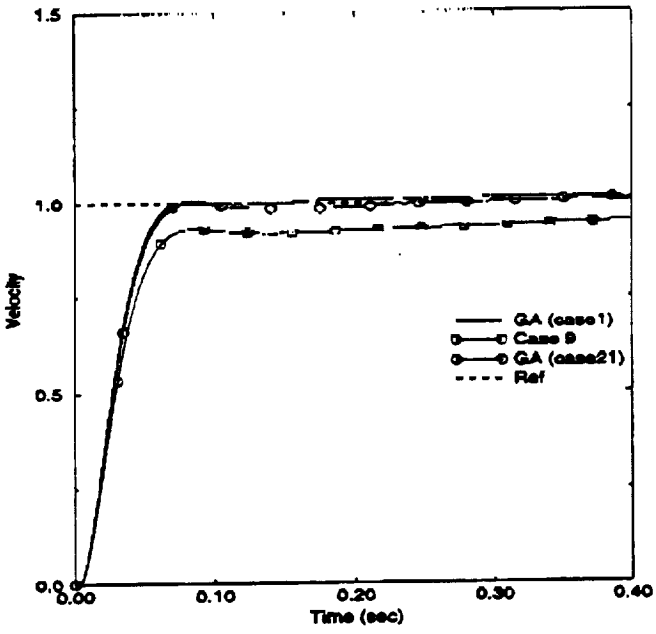

Figure 16. Simulation Result without Noise

follows.

- GA1 - GAs for PID gain control.
- GA2 - GAs for additional input control.
- SA1 - SA for PID gain control.
- SA2 - SA for additional input control.

According to the simulation results, GAs and SA show better performance than the original PID controller. Specifically, GAs for optimizing PID gain control performs the best.

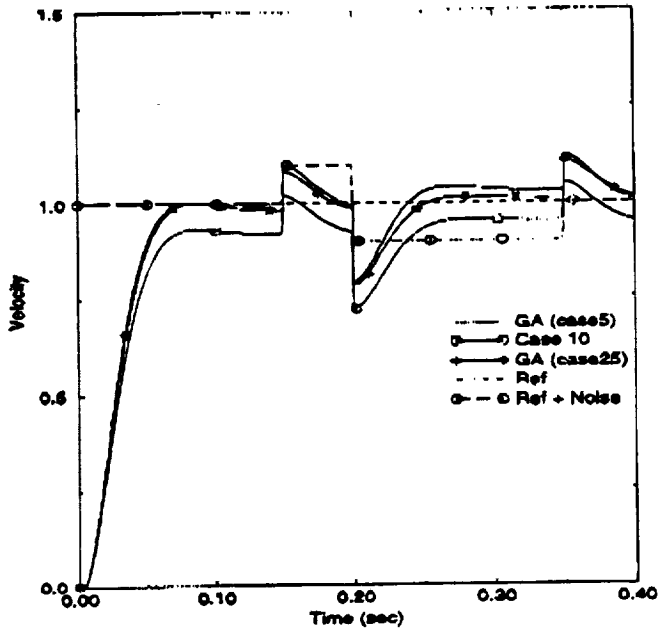

Figure 17. Simulation Result with Noise

VI. Conclusions

GMRAC techniques have been shown to outperform other adaptive control methods for the control of a PZT actuator system. Good control of PZT systems would enable them to be used in place of heavy, noisy motors in a variety of applications.

According to the simulation results, GAs and SA show better performance than a normal PID controller. Furthermore, GAs for PID gain control show the best result yielding a fast and stable response. GAs and a plant model were combined to form the GMRAC model, and it demonstrates good performance in dynamically controlling the system with or without noise.

References

[1] R. Das and D. Goldberg, "Discrete-time parameter estimation with genetic algorithms", in Proceedings of the 19th annual Pittsburgh Conference on Modeling and Simulation, Pittsburgh, PA., pp. 2391-2395, 1988.

[2] E.B. Ford, *Genetics and Adaptation*,Edward Arnold Ltd, London,1976.

[3] D. E. Goldberg,*Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, Reading, MA, 1989.

[4] J. H. Holland,*Adaptation in Natural and Artificial Systems*, Ann Arbor: University of Michigan Press, 1975.

[5] M. A. Lee and H. Takagi, "Integrating design stages of fuzzy systems using genetic algorithms", in Second IEEE International Conference on Fuzzy Systems, San Francisco, CA, pp. 612-617, 1993.

[6] D. Maclay and R. Dorey, "Applying genetic search techniques to drivetrain modeling", IEEE Control Systems, vol. 13, no. 2. pp. 50-55, 1993.

[7] Z. Michelewicz, et. al," Genetic Algorithms and optimal control problem", in Proceedings of the 29th Conference on Decision and Control, Honolulu, Hawaii, pp. 1664-1666, 1990.

[8] A. Owen," Grow your own programs," Computer Shopper, no.27,pp.191-193, 1990.

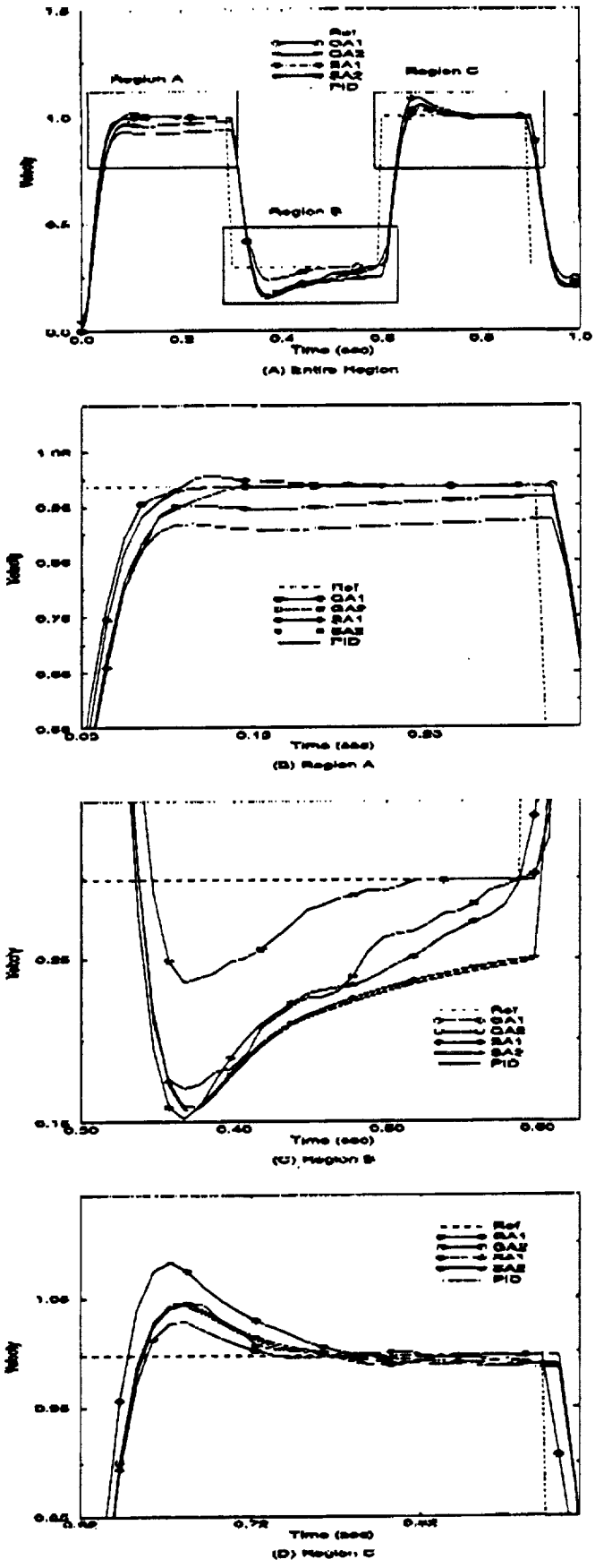

Figure 18. Simulation Result with Four Methods

[9] L. Porter and K. M. Passino, "Genetic Model Reference Adaptive Control", IEEE international Symposium on Intelligent Control, Columbus, OH, pp. 219-224, 1994.

[10] S. P. Sandford, W. S. Luck and W. W. Rohrbach, "Electromechanical Linear Actuators for Fourier Transform Spectrometers: a concept for extended range, high resolution, solid state actuators ", Applied Optics, Vol. 35, no. 16. pp. 2923-2926, 1996.